

# REE Radiation Fault Model: A Tool for Organizing and Communicating Radiation Test Data and Constructing COTS Based Spaceborn Computing Systems.

Raphael R. Some, REE Chief Engineer, [Raphael.R.Some@jpl.nasa.gov](mailto:Raphael.R.Some@jpl.nasa.gov)

Robert Ferraro, REE Project Manager, [Robert.Ferraro@jpl.nasa.gov](mailto:Robert.Ferraro@jpl.nasa.gov)

Jet Propulsion Laboratory

California Institute of Technology

4800 Oak Grove Drive

Pasadena, CA 91109

818-354-1902

**Abstract**—The growth in data rates of instruments on future NASA spacecraft continues to outstrip the improvement in communications bandwidth and processing capabilities of radiation-hardened computers. Sophisticated autonomous operations strategies will further increase the processing workload. Given the reductions in spacecraft size and available power, standard radiation hardened computing systems alone will not be able to address the requirements of future missions. The REE project was intended to overcome this obstacle by developing a COTS- based supercomputer suitable for use as a science and autonomy data processor in most space environments. This development required a detailed knowledge of system behavior in the presence of Single Event Effect (SEE) induced faults so that mitigation strategies could be designed to recover system level reliability while maintaining the COTS throughput advantage. The REE project has developed a suite of tools and a methodology for predicting SEU induced transient fault rates in a range of natural space environments from ground-based radiation testing of component parts. In this paper we provide an overview of this methodology and tool set with a concentration on the radiation fault model and its use in the REE system development methodology. Using test data reported elsewhere in this and other conferences, we predict upset rates for a particular COTS single board computer configuration in several space environments.

## TABLE OF CONTENTS

- I. INTRODUCTION
- II. REE PROJECT OVERVIEW
- III. METHODOLOGY AND TOOLS OVERVIEW
- IV. RADIATION FAULT MODEL
- V. CONCLUSIONS AND FUTURE WORK
- VI. ACKNOWLEDGEMENTS
- VII. REFERENCES

## I. INTRODUCTION

Over the past two years, NASA's Remote Exploration and Experimentation (REE) Project, has sponsored the development of test tools and techniques for characterizing the SEU effects of protons and heavy ions on complex state of the art COTS computers and components. REE sponsored tests have been performed on the PPC750, G4, AMD K7 and Intel P-III microprocessors, and associated peripheral support chips such as the Myrinet Lanai Processor [1,2,3]. These test results from organizations such as JPL, GSFC, and iRoC have been presented in previous RADECS and NSREC conferences, with the latest results being presented at this RADECS Workshop. Two aspects of this work stand out from previous radiation testing efforts:

A focus on high performance, complex, state of the art computer components, and

Significantly more detailed testing with respect to fault localization, with concomitant increase in the fidelity of fault types and rates.

This attention to the details of SEU effects is due to the peculiar needs of the REE program. We assume that the Spacecraft Control Computer and other critical avionics will be implemented in radiation hardened technologies, but that a high throughput onboard science data processing computer can be built out of COTS parts if the transient error modes can be predicted and mitigated. In the following sections we give an overview of the REE project and provide a perspective on where the radiation effects characterization and prediction fits in the overall project development strategy. We then explain the REE radiation fault model and show how this model is merged with experimentally derived fault effects data and environmental data, to predict fault rates, performance, reliability and availability for system

operation in various space environments. Finally, we provide some predictions of fault rates and reliability for a particular COTS single board computer configuration.

## II. REE PROJECT OVERVIEW

One of the objectives of the Remote Exploration and Experimentation (REE) Project is to demonstrate that COTS computing technology can be configured and operated reliably in many radiation environments important to NASA missions, including low Earth Orbit, Geosynchronous orbit, deep space, and the Mars planetary surface. To this end, the REE project has undertaken the testing and study of transient fault rates in COTS hardware, the effects of these faults on software components, and the development of Software-Implemented Fault Tolerance methodologies to mitigate the effects of radiation-induced errors in the applications that would run on such a platform in space. Due to the inherent Total Ionizing Dose (TID) tolerance of current CMOS technologies, REE's primary reliability concern is the detection and mitigation of SEE's without compromising high throughput, low power operation, and COTS compatibility.

REE's architecture of choice is a cluster computer. A cluster computer consists of interconnected stand-alone computers working together as a single integrated computing resource. Some of the salient characteristics of cluster computers are:

- Multiple high performance processors with local memory
- Fast network interconnects
- High-bandwidth/low-latency communication protocols
- A standard Operating System (OS) on each processor
- Access to shared mass storage
- A convenient parallel programming environment

Figure 1 shows the baseline REE architecture [4,5]. Each processing node consists of one or more microprocessors, local memory, a network interface to a multiply connected high speed network fabric, and an interface to a fault tolerant backdoor bus for housekeeping and diagnostics. These processing nodes communicate via messages to mass

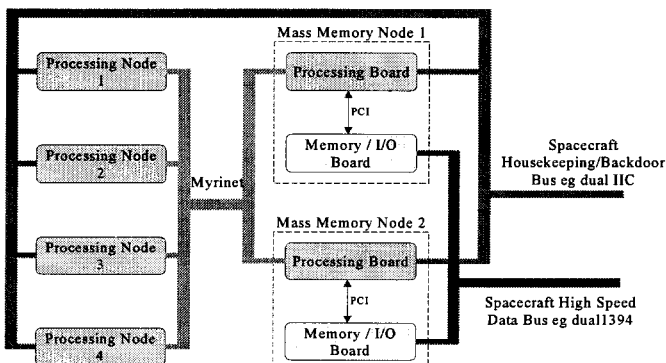


Figure 1: REE Cluster Computer Baseline Architecture

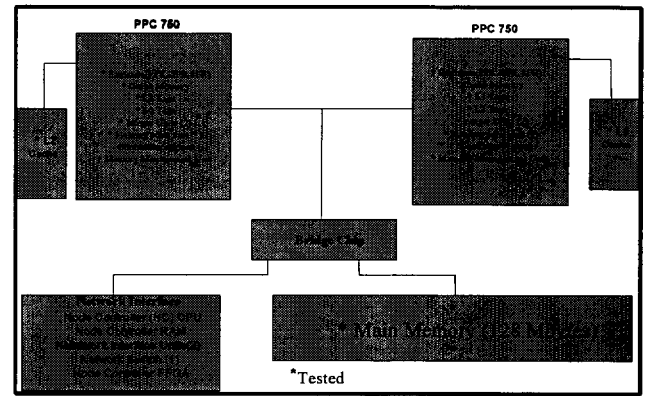


Figure 2: REE Testbed Dual PPC750 Node

memory nodes, and potentially to other types of special purpose processing nodes or intelligent instruments on the network. Message passing was explicitly chosen as the communications paradigm to provide fault containment boundaries around the elements on the network.

Figure 2 shows a processing node architecture developed jointly with AFRL. The dual PowerPC 750 microprocessors share access to main memory through the bridge chip, which also provides the PCI bus interface to other devices on the node. A Node Controller device, developed in collaboration with the Air Force Research Laboratory's Improved Space Architecture Concepts (ISAC) program, allows this standard single board computer architecture to interface in a fault tolerant manner with high-speed network fabrics (such as Myrinet). The Node Controller will also interface with the backdoor bus for external node control and diagnostics. Several of the components in this node architecture have been subjected to radiation testing (as noted in the figure). It is both noteworthy and serendipitous that advanced processor architectures are increasingly implementing low-level fault detection and protection mechanisms and some of these mechanisms are base-lined into the REE architecture. These include Single Error Correct Double Error Detect (SECCDED) Error Detection and Correction (EDAC) on local memories, parity protection on external caches, exception detection in Arithmetic and Logic Units (ALUs) and Memory Management Units (MMUs), and watchdog-configurable timers.

To make such a system reliable in the presence of random, transient faults, we need to be able to predict not only the fault rate expected in a particular space environment, but also the type (bit faults, functional interrupts, etc) and distribution (logic, caches, TLBs, arithmetic units, registers, etc.) of the expected faults. Knowing the expected rates, types, and relative probabilities allows us to tailor a detection and recovery strategy to the specific environment to maintain maximum throughput.

## III. METHODOLOGY AND TOOLS OVERVIEW

The REE project requires a means for trading off performance and power utilization versus reliability and

availability. The method must be generally applicable to alternative architectures and applications and, once developed, relatively straightforward to implement. Unlike traditional fault tolerant systems, a degree of unreliability or unavailability, i.e., .95 or .99 rather than .99999 is often an acceptable reliability figure for REE applications. On the other hand, it is imperative that the system fault behavior and reliability be accurately predictable. The mission system engineer must be able to 'dial in' a desired level of reliability and fault behavior based on mission phase and criticality. Thus, a methodology is required which will allow characterization and modeling of probabilistic system behavior, reliability, and availability under varying applications, environments, workloads, and operational scenarios.

Figure 3 shows the methodology and tool set developed for the REE project. It consists of a hierarchy of testing and modeling tools which combine radiation test data, anticipated environment conditions, and fault injection testing to predict system performance, availability and reliability during mission life. These tools and models can be adapted to specific processing and mass memory node architectures and

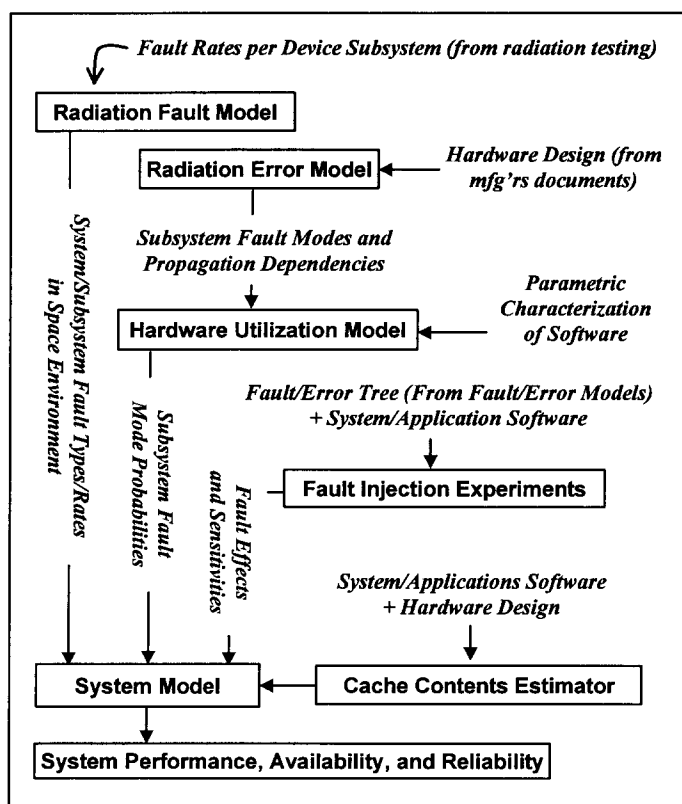


Figure 3: Overall REE Methodology Block Diagram

rely on minimal specific detailed knowledge of the circuits themselves. The final system performance evaluation incorporates software sensitivity to faults via detailed software testing using fault injection campaigns. Each major part of the methodology is explained below.

Radiation effects experiments are performed on the hardware components to determine subsystem level radiation sensitivities. Results for a processor, for example, may include fault rates for the L1 data cache, the L1 instruction cache, the General Purpose Registers (GPRs), the Floating Point Registers (FPRs), the MMU, etc. It is advantageous to have very detailed test data that distinguishes fault rates for the various functional components of a device, but the methodology can proceed with gross fault rates as well.

The results of the radiation experiments are used to develop a system radiation fault model. This model projects the radiation test fault rates of all the components in the system to their expected fault rates in a specific radiation environment by adjusting the rates for the specific environment flux and particle spectrum. It then combines these adjusted component rates together into a system level prediction of the fault rates that will occur in a given radiation environment (e.g., Low Earth Orbit, Geosynchronous Orbit, Deep space, Solar Flare, etc.). The model is hierarchical and easily configurable to provide the average number of faults per unit time at the functional block, subsystem, and system levels.

The Error Model predicts, by analysis of the system hardware, the types of errors that can arise as a result of an SEU occurring in a given subsystem. This step is necessary because individual faults in components may or may not result in an observable error in function. This is done without resorting to the use of proprietary design data and, as is true throughout the design methodology, can be effectively accomplished with publicly available component data. The validation of this ability to avoid the use of proprietary data is crucial, as system designers almost never have this data. In addition, the error model is key to devising fault detection/localization strategies for radiation testing and for coordination between system designers and radiation test engineers.

Essentially, the process of generating the error model is one of listing all possible faults and then, by analysis, propagating each fault through the hardware to the first point at which it impacts software or system operation. The emphasis of this effort is on subsystems into which faults cannot be directly injected with Software Implemented Fault Injection (SWIFI). Thus, it is not necessary to trace every possible error resulting from a general-purpose register bit flip. It is however, necessary to list all the possible outcomes of SEUs in MMU and cache address translation registers, cache tag rams, etc.

The Hardware Utilization Model provides a means for determining the software dependent probabilistic error propagation statistics and the method by which Software Implemented Fault Injection (SWIFI) techniques can be used to emulate the effects of the underlying errors enumerated in the Error Model. Radiation induced faults that do manifest as transient errors may never become effective because the software executing on the hardware only utilizes a small fraction of the hardware at any given time. It is often the case

that an error in a microprocessor cache or register gets flushed by incoming data without ever entering the computing sequence. This is a function of the utilization of registers, caches, and other devices in the system by the operating system and the applications being executed, and so needs to be folded into the reliability analysis.

Fault injection campaigns are designed to provide fault/error sensitivities of the system and application software. The campaigns are conducted on the operational system and are designed to diagnose the sensitivity of the software to specific error types (register bit flips, instruction cache errors, data cache errors, table lookaside buffer errors, etc.). An automated campaign of tens of thousands to hundreds of thousands of fault injections is conducted on the operating software, and the result of each fault injection is analyzed to determine the effects of the faults (e.g., system crash/hang, incorrect result, no apparent effect) and their associated probabilities. Empirically determining fault sensitivities of various sections of code, types of codes and the hardware used in executing the codes, allows us to then develop a systematic mapping of subsystem susceptibilities and fault effects which can then be used as the basis for developing system design guidelines, coding guidelines, hardware design guidelines and fault detection and mitigation strategies and techniques.

The Cache Contents Estimator (CCE) is used to deal with the current inability of SWIFI techniques to inject bit flip faults into the processor's cache memories. It is, in effect, a special case of the Error Model/Hardware Utilization Model explained above. Faults are injected into an application's instruction, data, heap, and stack segments in main memory to determine the fault behavior statistics of each type of error. The CCE predicts how much (and which portions) of each of these segments will be in the cache at any given time. The final error rate for each of these segments in cache is proportional to its size, and time of residence in unprotected L1 Cache. (For the latest version of the G4 processor, which incorporates parity protection on internal caches as well as state registers, this tool is used to predict system exception rates and resultant impact on real time performance.)

Finally, the system reliability and performance model is constructed using knowledge of the system architecture, predictions from the fault model, the results of the fault injection experiments and the CCE results. The model predicts the system's reliability and performance in a given radiation environment. It can be used during system development to identify appropriate system architectures and fault tolerance strategies. During fielded operation, the model can be used to predict the system's behavior in changing circumstances and modify it as appropriate (e.g., increase checkpointing frequency, uplink fault-tolerant linear algebra libraries). Once the basic system model has been created and validated, it is relatively straightforward to input application software specific fault behavior statistics, input the mission environmental parameters, and predict system fault behavior and reliability for a range of fault tolerance

techniques. Used this way, it also provides an early quick look resource for mission development.

Two system models were developed on the REE project. A Semi-Markov type model which represented the possible system states, the events causing transition between the states and the probabilities or rates of occurrence of these transitional events in a given environment and with a given operational system load was first developed to estimate system reliability and availability in deep space, low earth orbit, and on the surface of Mars. Subsequently, a second model was developed using similar techniques, but incorporating performance estimation as well as reliability and availability. Using this second, more complex model, it is possible to perform system configuration studies leading to optimization of performance versus reliability and availability under a broad range of operating conditions and environments.

In summary, the overall flow of this analysis method is to:

- Get detailed SEU rate and type data for each system and subsystem element

- Analyze the hardware architecture to determine how SEU type faults propagate and are manifest as errors in those subsystems into which we can not directly inject faults

- Develop fault/error injection campaigns to determine the impact of these hardware faults on the software and of the resultant software errors on the system

- Build a model of the system into which is input the fault rates and the system fault behavior and which then provides an estimate of the system performance, reliability and availability.

The detailed information on the software sensitivity to faults gathered during the fault injection experiments is used to select and implement fault detection and mitigation strategies and techniques. These techniques are then incorporated into the system model, and system performance and reliability with these new techniques is estimated. This updated analysis allows the designer to evaluate each technique for effectiveness without actually implementing and testing each one. For example, checkpointing the entire application process at some fixed interval with the intention of rolling back in the event of an error might seem like an appropriate fault mitigation strategy. However, the system model could indicate that this approach, compared to simply repeating the entire data frame calculation, has a significant performance penalty for only a modest gain in reliability. These fault protection design trades can thus be evaluated without actually coding them. Only those deemed optimal are then coded and tested with fault injection campaigns to verify or recalibrate the model.

In this development methodology, a key element is the radiation fault model. It is this model which provides a common communication medium between radiation

physicists, hardware and software engineers, reliability and system analysts, and system engineers. The development of the REE radiation fault model played a crucial role in determining the design approach use for the radiation test bed and for the subsequent radiation data analysis, as well as the design of the REE hardware system and of the error model and fault injection campaign strategy. The radiation fault model was also a key input in both the construction and execution of the REE system model. Our experience was that this model was an invaluable tool for system design, test, modeling and verification.

In a previous paper [6], the details of applying this system modeling methodology were explained using three single board computers in a TMR configuration. A test bed was constructed to evaluate several fault tolerance approaches. Fault rates derived from the radiation fault model were used in the manner described above, and system reliability was evaluated for a number of space environment conditions. Details of the fault inject campaigns, their results, and the methodology for combining those results with the radiation fault model predictions were presented. The interested Reader is referred to this paper for more details.

#### IV. RADIATION FAULT MODEL

The REE Radiation Fault Model is a hierarchically organized tool in which, at the lowest level, the system is represented as a collection of simple functional blocks such as register sets, MMUs, cache control registers, etc. These functional blocks are grouped to form subsystems or “nodes”. Systems are configured by selecting nodes and any required miscellaneous hardware.

The model predicts fault rates at the functional elements for a specified environment and propagates these faults to the higher levels of the model, accumulating them at each level to predict fault rates and types at the node and system levels.

The model also has another use: it functions as a handy communications medium between system designers and radiation test engineers. In general, these two groups speak in different vocabularies. The model forces a common language, which is meaningful to both. As such, it also helps organize and prioritize the work each organization is performing. By providing a parametric view of the system, the model also allows prediction of fault rates in untested components or functional blocks and early prediction of system reliability.

CPU design - IBM PowerPC 750					
Latch fault rate (LFR)	Source	Param	# Latches	LFR	LFR*Faults
Totals:			919875		0.72781438
Chip area, mm <sup>2</sup>		825			
Address bus width, bits		32			
Data bus width, bits		32			
GP Register/Instruction/Address width, bits		32			
FP Register width, bits		64			
Width of Virtual Address		52			
Width of Real (Physical) Address		32			
Width of Page offset		12			
Width of Real Page Number		40			
Number of General Purpose (GP) registers		32	1024	9.54E-07	0.0003769
Number of floating point (FP) registers		32	2048	9.54E-07	0.00195379
Number of special registers (CR, LR, CTR, XER, FPSCR)		5	160	9.54E-07	0.00015254
Number of Program Counters (PC)		1	32	9.54E-07	3.0528E-05
Number of Control/Status Registers (CSR)		42	1344	9.54E-07	0.00129218
Number of debugging registers (decr, watchaddr, watchinstr)		3	96	9.54E-07	9.1584E-05
Number of addressing registers (in BLU)		4	256	9.54E-07	0.0024422
Number of latches holding current instruction		29	928	9.54E-07	0.00089531
Number of register rename buffers (6 GP, 6 FP, 3 CSR)		15	672	9.54E-07	0.00064109
No. of Branch Target Instruction Cache entries		64	2048	9.54E-07	0.00195379
No. of Branch History Table entries		512	1024	9.54E-07	0.0003769
Number of MMU entries/TLB		128		9.54E-07	0
Tag bits per TLB entry (dirty, protected, read-only, volatile, write-back)		10		9.54E-07	0
Width of MMU TLB entry, bits		70		9.54E-07	0
Number of MMU TLB's		2	17920	9.54E-07	0.01709698
Latches/BAT		64		9.54E-07	0
Number of I/D BATs (defined as SPRs/shadowed)		16	1024	9.54E-07	0.0003769
Number of MMU mem segment registers, VSID-SLB		17	408	9.54E-07	0.00039823

Figure 4: PPC750 Input Data Sheet

Several types of parametric data are entered for each functional block. If the number and types of registers, gates, and memory elements are known or can be estimated from available data, these are entered along with proton and heavy ion SEU sensitivity data for those circuit types. Where this data is unavailable, I/O level fault rates for the block as a whole are used. In addition, some functional elements have an associated SEFI rate, i.e., a rate of untraceable or un-localizable fault, which results in functional failure. SEFI faults generally include clock distribution circuits and mode control logic. Scaling factors are included for estimation of next generation process technology and circuit complexity effects as well as for “safety margin”.

Figure 4 shows a portion of an input data sheet for the PPC750 processor. On the left side of the sheet are the architectural parameters of interest for each functional block such as the number of general-purpose registers and the total latch count for that register set. The right side lists the per latch fault rates for each functional block and the total fault rates in the environment of interest. Not shown in the figure, due to size constraints, are the other circuit and fault types, such as gate and memory fault rates. The complete data sheet contains as much detailed architecture information as is available for the processor. The overall fault rate is determined from the measured latch fault rate and the number of latches in each of the device functional components.

Node-Level design					
	Count	Margin	Latch Faults/hr	Gate Faults/hr	Total Faults/hr
Totals per node:			6.95	0.05	6.99
Node CPU's per node	2		5.31	0.04	5.36
Node Controller (NC) CPU	1	1.5	0.71	0.00	0.71
Node Controller RAM	1	3	0.12	0.00	0.12
Network Interface Units (NIU) per node	2	3	0.35	0.00	0.35
Number of Network Switches per node	1	3	0.16	0.00	0.16
Bus controller (PCI)	1	3	0.13	0.00	0.13
Misc (watchdog, clock, EEPROM, PHRC)	1	3	0.02	0.00	0.02
Node Controller FPGA	1	3	0.14	0.00	0.14

Figure: 5 Node Level Input Data Sheet

Daily Average Single-Bit Cache Upset Rates (Upsets/Day) Detected by Parity Check											
Environment	Interplanetary Space (near Earth)				High-incline 600km-98° Earth Orbit				600km-28°	Mars Surface	
Shielding	100 MI (Al)				100 MI (Al)				100 MI (Al)	NA	
Flare Status	No Flare		Design Case Flare		No Flare		Design Case Flare		NA	NA	
Solar Minimum/Maximum	Solar Min	Solar Max	Solar Min	Solar Max	Solar Min	Solar Max	Solar Min	Solar Max	NA	NA	
L2 Cache Parity-Check 256K	2.36E+06	2.643	0.661	4958.643	4956.661	1.510	1.020	1029.668	1029.177	1.751	0.170
PPC750 System Total		2.643	0.661	4958.643	4956.661	1.510	1.020	1029.668	1029.177	1.751	0.170
L2 Cache Parity-Check 256K	2.36E+06	1.180	0.297	2313.980	2313.097	0.727	0.510	467.592	467.374	0.871	0.076
PPC7400 System Total		1.180	0.297	2313.980	2313.097	0.727	0.510	467.592	467.374	0.871	0.076
PPC7450 L1 Caches	5.65E+05	0.283	0.071	553.983	553.771	0.174	0.122	111.945	111.893	0.208	0.018
PPC7450 L2 Cache & Tag	2.47E+06	1.235	0.311	2421.835	2420.911	0.761	0.534	489.386	489.159	0.911	0.079
PPC7450 System Total		1.518	0.382	2975.818	2974.682	0.935	0.656	601.331	601.051	1.120	0.097

Figure 6: PC750 vs G4 Cache Parity Upsets Detected Per Day

Figure 5 shows an input sheet for the node illustrated in Figure 2. It enumerates the discrete components of the node level system, and combines the fault rate predictions for each of these components into a rate prediction for the entire node. A “margin” figure is included to allow varying levels of conservatism in the output, as some data used in the individual discrete component fault models are estimates (e.g. – number of latches in an MMU TLB). These margins can be used to assert the uncertainties associated with devices that have not actually been radiation tested, but have estimated rates based on similar device complexity and fabrication technologies. Allowing the user to provide an explicit safety margin in the calculations makes it clearly evident and prevents the user from embedding “hidden safety margins” in the circuit and functional block counts. The right side of the sheet shows the accumulated latch and gate fault rates and the total fault rates per functional block. As can be seen, the testing to date has not shown a significant gate circuit fault rate. This was somewhat surprising at the time, but has been satisfactorily explained. The model, however, retains a gate circuit fault rate for future use.

Using the methodology described above, we have performed an analysis of cache upset rates for two complex microprocessors for which we have detailed radiation test data. Figure 6 is an output sheet of the radiation models comparing fault rates between PPC750 and G4 (PPC7450) processors in various environments. This sheet is one of may

such outputs which have been extremely useful in the design of the REE system. Note that the PPC 750 does not support parity on the internal memories and latches, while the G4 processor provides parity on all its internal L1 cache and cache control registers. These rates are predicted based on the radiation environment attenuated by 100 mils of aluminum shielding in four space environments: Geosynchronous, Low Earth Polar, Low Earth moderate inclination, and the surface of Mars. In two cases, the rates are further adjusted to account for solar cycle variability and the presence or absence of a “Design Case Solar Flare”. This flare is taken to be of extremely high intensity – but not a theoretical worst-case scenario. The rates are orbit averaged, and in the flare case, are at peak flare intensity. The magnetosphere effectively shields out the effects of a solar flare for the low inclination orbit calculation, so there is no change in rate for that case. Solar Min rates are presented in that case as well.

We emphasize that these are rates in upsets per day for the processor/cache system as a whole that would result in a parity violation if they were actually read out in the course of computation. These rates can also be used to determine the probability of an undetectable double bit error in these caches as well as set a cache flushing strategy to minimize the probability of such an occurrence. This output is also useful for the design, selection, and tailoring of specific software implemented fault tolerance mechanisms for a given mission environment and operational scenario.

	0.04	0.01	28.46	28.43	0.05	0.02	1.89	31.27	0.00	0.03
Node CPU (w/o Caches)	0.04	0.01	28.46	28.43	0.05	0.02	1.89	31.27	0.00	0.03
L1 Cache	0.05	0.01	37.40	37.37	0.06	0.02	7.54	5.86	0.00	0.04
RAM per node CPU	0.01	0.00	7.94	7.93	0.01	0.00	1.67	7.67	0.00	0.01
L2 Cache	0.00	0.00	1.03	1.03	0.00	0.00	0.22	1.67	0.00	0.00
Node Controller CPU	0.03	0.01	22.10	22.08	0.04	0.01	4.61	0.22	0.00	0.02
Node Controller Ram	0.01	0.00	4.30	4.20	0.01	0.00	0.88	4.60	0.00	0.00
Network Interface Units per node(2)	0.01	0.00	6.87	6.86	0.01	0.01	1.14	0.88	0.00	0.01
Network Switch Per Node(1)	0.01	0.00	3.17	3.17	0.00	0.00	0.53	1.34	0.00	0.01
PCI Bus Controller	0.00	0.00	2.58	2.58	0.00	0.00	0.43	0.52	0.00	0.00
Node Controller FPGA	0.00	0.00	1.18	1.18	0.00	0.00	0.24	6.43	0.00	0.00
watchdog,clock,EEPROM,PIRC	0.00	0.00	2.63	2.63	0.00	0.00	0.44	0.24	0.00	0.00
Single CPU	0.10	0.03	74.83	74.76	0.13	0.05	15.66	15.64	0.01	0.06
Per Node	0.25	0.07	192.40	192.21	0.33	0.12	39.27	39.53	0.02	0.11
Per System (20 nodes)	4.97	1.33	3852.98	3849.34	6.27	2.39	792.71	791.57	0.34	1.14

Figure 7: System Summary

Figure 7 carries this analysis to the full node level for the processing node outlined in Fig 2. On the left side of the sheet are listed the major system elements. The columns to the right show the aggregate fault rates for each of these elements in the current model environments. (Table entries were rounded to two decimal places for presentation purposes, after carrying all significant digits through the calculation.) The bottom two rows show the node and system level fault rates. This system summary shows aggregated average fault rates in the same environments covered in Fig 6. Note that here the scale is faults per hour, and covers a dual processor node system and its attendant components. Not all of the components were actually radiation tested. Estimated fault rates (with margins) were derived for those components by estimating gate and latch counts, and applying rates measured on different devices that use the same fabrication technology. These untested components do not contribute a major component to the system fault rate, which is dominated by the microprocessors themselves. The fault rate for the individual DRAM memories is substantial, but does not contribute significantly to this system rate because the memory structure is EDAC protected. Only double bit exceptions are factored into this calculation. This type of output sheet provides a handy summary of the model and readily points to areas of weakness in the system design.

## V. CONCLUSIONS AND FUTURE WORK

The REE project, over the past 2 years, has developed a methodology and tool set for designing highly reliable systems using COTS components in radiation environments. A key element of the methodology is the Radiation Fault Model, which provides a tool for easily collating and organizing radiation test data and for communicating radiation effects data and implications between radiation physicists and system designers. The tool is, at present, implemented in a spreadsheet and would benefit from a user-friendly front end and integration with other radiation effects

test tools developed by JPL, GSFC, and iRoC on the REE project. Orthogonalization of the radiation test data inputs should also be done in the future. The tool is, however, useful in its current state and has been used on the REE project for preliminary system design studies.

Preliminary results, as shown in Figure 7, strongly imply the applicability of COTS high end computing to some space environments when appropriately used in a fault tolerant system. The Martian surface stands out as an extremely benign environment where a COTS computer would enable significant increases in science return from rovers and landers.

## VI. ACKNOWLEDGEMENTS

This work is derived from a model for the PPC750 originally developed by John Beahan at JPL. Arbi Karapetian at JPL assisted in the expansion of this model to a more general form, and Won Kim at JPL performed the analysis of the Motorola G4 processor.

This work was performed at the Jet Propulsion Laboratory, California Institute of Technology under contract with the National Aeronautics and Space Administration. The Remote Exploration and Experimentation (REE) Project is part of NASA's High Performance Computing and Communications Program, and was supported by NASA's Space Science Enterprise.

## VII. REFERENCES

- [1] Single-Event Upset in the PowerPC750 Microprocessor, Gary M. Swift, Steven M. Guertin, Farhad Farmanesh, Farokh Irom, D.G. Millward, IEEE Transactions on Nuclear Science 48(6) pp 18202-1827 2001
- [2] Single Event Testing Using Heavy Ion Irradiation Through Thick Layers of Material, G. M Swift, D. G.

Millward, H. L. Clark, Proceedings of the RADECS 2001 conference

[3] Single Event Upsets in Commercial Silicon on Insulator PPC Microprocessors, F. Irom, F. Farmanesh, G. M. Swift, A. H. Johnston, D. G. Millward, 2002 Nuclear and space radiation effects conference Phoenix Arizona, July 2002; to be submitted for publication in IEEE Transactions on Nuclear Science Dec. 2002.

[4] REE: A COTS-Based Fault Tolerant Parallel Processing Supercomputer for Spacecraft Onboard Scientific Data Analysis, R. Some, D. Ngo, Proceedings of the Digital Avionics Systems Conference (DASC) 1999.

[5] Detailed Radiation Fault Modeling of the Remote Exploration and Experimentation (REE) First Generation Testbed Architecture, J. Beahan, R. Some. R. Ferraro, L. Edmunds, A. Johnston, D. Katz, Proceedings of the IEEE Aerospace Conference 2000

[6] Fault Injection Experiment Results in Space borne Parallel Application Programs, Raphael R. Some, Won S. Kim, Garen Khanoyan, Leslie Callum, Anil Agrawal, John J. Beahan, Arshaluys Shamilian, Allen Nikora, Proceedings of the IEEE Aerospace Conference 2002